NLP -CS6320 Final Project Extra credit

Saivikas Meda sxm190011

EMOTION DETECTION

Github: https://github.com/saivikasmeda/EmotionDection.git

Description

1. **Data Preprocessing - The complete text is converted into lower case and links, stopping words and special characters are removed. The emojis are converted into text.**
2. After preprocessing the json objects are converted in such a format that each emotion is attached to text as a different label. So, let's say if a text has 3 emotions then three different rows are created each time with an emotion as a label to it.

```
In [7]: dataset.cleaned_data.head()
```

Out[7]:

	label	text
0	anger	e answering question criticism individual refe...
1	disgust	e answering question criticism individual refe...
2	pessimism	e answering question criticism individual refe...
3	anger	'm going start today's discussion thread perso...
4	anticipation	'm going start today's discussion thread perso...

3. **Word vector representation** - We cannot send the words directly to model. We've converted words into vectors of a specific dimension. I've used an inbuilt tokenizer model provided by keras. tensorflow.keras.preprocessing.
4. I've spilt training data into training and validation data. I've used 80% of data for training and 20% for validation.
5. The model that I'm using is **LSTM + CNN (Convolutional Neural Network).** The input to LSTM is a 750 dimension vector that is created from words. This is now forwarded to spatial dropout. This dropout probability of 0.2 is used - the probability of dropping out inputs to a layer is to regularize the neural network. Now the embedded input is sent to the LSTM. The output from this is sent to CNN and this network is 1D as the data is a 1D text document. Average and Max Pooling is performed with softmax activation function. This model is trained for 50 epochs.

```
In [13]: model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
         model.summary()
```

```
Model: "model"
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================================
input_1 (InputLayer)         [(None, 150)]             0
_________________________________________________________________________________________________
embedding (Embedding)        (None, 150, 750)          7500000     input_1[0][0]
_________________________________________________________________________________________________
spatial_dropout1d (SpatialDropo (None, 150, 750)       0           embedding[0][0]
_________________________________________________________________________________________________
bidirectional (Bidirectional)   (None, 150, 260)       916240      spatial_dropout1d[0][0]
_________________________________________________________________________________________________
conv1d (Conv1D)              (None, 148, 64)           49984       bidirectional[0][0]
_________________________________________________________________________________________________
global_average_pooling1d (Globa (None, 64)             0           conv1d[0][0]
_________________________________________________________________________________________________
global_max_pooling1d (GlobalMax (None, 64)             0           conv1d[0][0]
_________________________________________________________________________________________________
concatenate (Concatenate)    (None, 128)               0           global_average_pooling1d[0][0]
                                                                   global_max_pooling1d[0][0]
_________________________________________________________________________________________________
dense (Dense)                (None, 12)                1548        concatenate[0][0]
=================================================================================================
Total params: 8,467,772
Trainable params: 8,467,772
Non-trainable params: 0
_________________________________________________________________________________________________
```

6. The trained model weights are now saved in the “model_weights.h5” file.

```
Epoch 11/20
3568/3568 [==============================] - 57s 16ms/sample - loss: 1.3659 - acc: 0.2671 - val_loss: 4.4405 - val_
acc: 0.0176
Epoch 12/20
3568/3568 [==============================] - 56s 16ms/sample - loss: 1.3581 - acc: 0.2713 - val_loss: 4.4884 - val_
acc: 0.0202
Epoch 13/20
3568/3568 [==============================] - 56s 16ms/sample - loss: 1.3525 - acc: 0.2691 - val_loss: 4.5269 - val_
acc: 0.0208
Epoch 14/20
3568/3568 [==============================] - 58s 16ms/sample - loss: 1.3463 - acc: 0.2744 - val_loss: 4.5451 - val_
acc: 0.0234
Epoch 15/20
3568/3568 [==============================] - 57s 16ms/sample - loss: 1.3426 - acc: 0.2721 - val_loss: 4.6009 - val_
acc: 0.0195
Epoch 16/20
3568/3568 [==============================] - 56s 16ms/sample - loss: 1.3415 - acc: 0.2668 - val_loss: 4.6197 - val_
acc: 0.0176
Epoch 17/20
3568/3568 [==============================] - 59s 17ms/sample - loss: 1.3323 - acc: 0.2693 - val_loss: 4.7051 - val_
acc: 0.0163
Epoch 18/20
3568/3568 [==============================] - 54s 15ms/sample - loss: 1.3370 - acc: 0.2657 - val_loss: 4.6514 - val_
acc: 0.0202
Epoch 19/20
3568/3568 [==============================] - 54s 15ms/sample - loss: 1.3384 - acc: 0.2864 - val_loss: 4.6242 - val_
acc: 0.0195
Epoch 20/20
3568/3568 [==============================] - 54s 15ms/sample - loss: 1.3333 - acc: 0.3013 - val_loss: 4.6999 - val_
acc: 0.0182
```

After 45 epochs the loss and acc were changing much. So i just stopped at 50 epochs.

7. We’ve to run a test_model.py file to test the model and detect the emotions for the text provided using the trained LSTM+CNN model. The test data is first preprocessed as mentioned earlier. Even the test data is converted into vector

presentation and passed to LSTM. The output of CNN provided the probability of each emotion occuring. Higher the probability likely the emotion in the text.

8. I'm also displaying the average probability of each emotion throughout the data test. This value is to know how data is related to a particular category. For instance i'm displaying text id and emotions that are detected:

```
fhlsd2m :
anticipation; fear; neutral


fka56ha :
fear; neutral; sadness


finry2a :
anger; disgust; pessimism; sadness


fl4e0zp :
anticipation; disgust; fear; optimism; pessimism; sadness


fkasyqk :
anger; anticipation; disgust; fear; pessimism
```

Metrics Sore :
F1score : 76%
Train_accurancy – 0.378
Test_accuracy – 0.189

For f1 score initially i've divided train data into 80-20% for training and testing purposes. Using this data i've calculated f1Score. Later with the provided data we can't find f1score as data doesn't contain emotions true values.

9. The softmax function will provide the probability for each emotion. I've taken an avg of each emotion in the entire test data set. For each text if the emotion probability is more than average value that particular emotion is detected in the text.
10. I've tried using Signmod and relu as activation functions but even these are resulting in probability. Softmax function provides a wide range in probability which helps in detecting the emotion for text. So, I ended up using softmax.

11. Few points to before getting into details. The train data file should be named as nlp_train.json and the test data file should be named as nlp_test.json. These json objects should contain “id”,”body” and “emotion” as keys.